Computing Valid Intervals for Collections of Activities with Shared States and Resources

Russell Knight, Gregg Rabideau, and Steve Chien
Jet Propulsion Lab, M/S 126-347
California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109-8099
firstname.lastname@jpl.nasa.gov

Abstract

When scheduling a collection of activities, it is often useful to calculate the valid intervals for the collection with respect to shared resources and states. The constraints of these activities on shared resources and states need to be analyzed to avoid miscalculations due to interactions between the constraints. For shared resources, a combined profile is typically generated and used to compute the valid intervals. We present a technique for generating a combined profile for shared state constraints, which is subsequently used to compute valid intervals. We present empirical evidence indicating that our technique improves performance of our planner on real and synthetic problems when compared to the performance of the same planner using more naïve techniques.

Introduction

Although local, heuristic problem-solving methods have been used with considerable effectiveness in solving some large-scale problems (Lin, S. and Kernighan, B. 1973, Zweben, M., Daun, B., Davis, E., and Deale, M., 1994), these methods are stymied by interdependent activities. This is because in order to resolve a problem with one of the coupled activities, the plan is likely to require modification of all of the clustered activities. To the local search, when changing a single one of the activities, it appears that resolution of a single conflict has caused a large number of new conflicts that may take considerable effort to resolve. Thus, the local search method tends to get stuck in a local minimum.

This paper describes an approach to solving the problem of scheduling interdependent sets of activities. In this approach, we compute a combined profile that encapsulates the requirements and effects of the block of activities on shared states and resources. This combined profile is then used to determine legal placements for the complete set of activities allowable in the current plan.

This problem is an important aspect of solving combined planning and scheduling problems. In many approaches to combined planning and scheduling, one alternates between finding activities to satisfy pre- and post-conditions (planning) and finding temporal assignments and resources for those activities (scheduling). Complex activity placement is also an important component of many scheduling problems, as finding temporal assignments for complex activities can be computationally challenging.

This work advances the approach of moving collections of activities whose temporal relationships among themselves are fixed. It also is related to job-centered scheduling approaches and tabu-search approaches, where similar techniques are used to schedule collections of activities. We proceed by describing our motivation, defining the problem, and describing the solution. Finally, we present empirical evidence in favor of our technique.

Motivation

We wish to perform scheduling of collections of inter-related activities that use shared states and resources (for whatever reason), as opposed to scheduling individual activities. We assume that the temporal relationships between members of the collection are fixed. Even the simplest of approaches require that we know the valid intervals for a given collection. We discover that naïve intersection of valid intervals for single activities in the collection render both false positives (intervals returned as being valid in fact cause constraint violations) and false negatives (intervals returned as causing constraint violations are in fact valid) with respect to the whole. This can only be due to interactions between the activities; or, more specifically, interactions between the constraints on shared states and resources.

For example, consider a pair of activities that affect a battery (see Figure 1). The first activity a_1 uses 10 amp-minutes, while the second activity a_2 restores 10 amp-minutes. If we schedule a_1 individually, we find no intervals that will not cause an over-subscription of the battery, because activity a_3 has already fully depleted the battery by the end of the current schedule. But, if we schedule these together, we find placements that are valid.

The positive effect a_2 has on the schedule makes up for a_1's usage. We wish to handle this sort of constraint-interaction for shared states and resources.

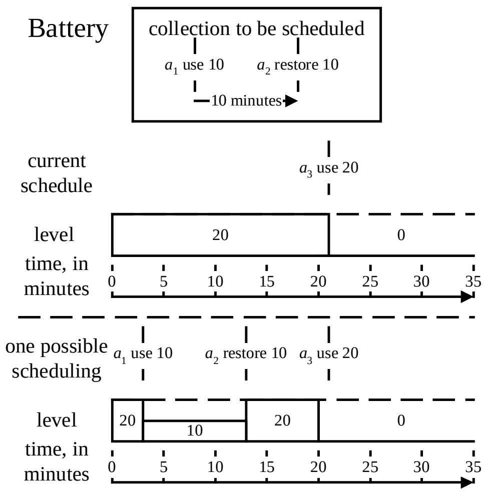

Figure 1 Battery interaction example

Problem and Definitions

Given a schedule (S) and a collection of activities (Q) to be scheduled whose temporal relationships among themselves are fixed, find the interval set (I) representing assignments of the start-time of the earliest activity in the collection that violate no constraints. A few definitions are in order—

Schedule: consists of the set of activities that have been scheduled (A) and the set of timelines representing the shared states and resources (T).
Activity: consists of a start-time (s), an end-time (e), and a set of timeline constraints or reservations that represent the constraints of an activity with respect to shared states and resources (R). R is conjunctive, therefore we do not deal with multiple resources that could satisfy a single constraint. For simplicity, we assume that times are integers. We also assume that $s < e$.
Interval: a pair of integers $(a, b) \mid a \leq b$.
Interval Set: a set I of intervals where no two intervals overlap. Specifically, $\forall(a_1, b_1) \in I, \forall(a_2, b_2) \in I \mid (a_1, b_1) \neq (a_2, b_2), \neg(a_1 \leq a_2 \leq b_1) \cap \neg(a_1 \leq b_2 \leq b_1) \cap \neg(a_2 \leq a_1 \leq b_2)$.
Horizon: the interval that all time-points are contained within.
Timeline: the representation of a shared state or resource over time. All timelines consist of a set of value units (U) that represent the value of the timeline for all time-points in the horizon. A value unit is a pair (t, v) where t is the start-time of the unit and v is the value. So, U consists of at least one unit (the unit that begins at the beginning of the horizon), and all units taken in order of time represent the value of the shared state or resource over time. All timelines also consist of a set of global constraints. Global constraints and values differ according to the type of timeline.
Depletable Resource Timeline: a timeline representing a resource that is added to or removed from over time. An example is a battery, where usage depletes the battery and recharging restores the battery. Global constraints include the minimum level (min), the maximum level (max), and the default or starting value (d). For our discussion, all values will be integers.
Non-Depletable Resource Timeline: a timeline representing a resource that is used for a period of time and then relinquished back at the end of the usage. An example is a power bus, where usage takes up some of the capacity of the bus, and the cessation of usage restores the capacity. Obviously, a non-depletable timeline can be represented by a depletable timeline with a depleting reservation followed by a renewing reservation (reservations are described later), but we include these because the semantics of their reservations aids in describing the semantics of transformed depletable reservations. As with depletable timelines, global constraints include the minimum level (min), the maximum level (max), and the default value (d).
State Timeline: a timeline representing a state that can be either changed or required by reservations. Values are symbols; we use strings. Global constraints consist of a transition graph G whose nodes represent allowed values and whose edges represent allowed consecutive transitions from one value to the next, and a default value (d).
Reservation: a constraint on a shared state or resource for a specific interval to a specific value. Since reservations are part and parcel with activities, reservations inherit their start- and end-times (s, e) from the activity containing them. A reservation also consists of a reference to a timeline. The type of timeline that the reservation constrains determines the type of the reservation.
Depletable Reservation: consists of a start-time (s, the end-time is ignored) and a value (v). v is an integer representing the amount of capacity to be used by the reservation. The value of any point on such a timeline is the sum of all reservations at the time-point and previous, as well as the default value. See Figure 2. Note that the timeline models a value over time. The reservations are labeled with their values.

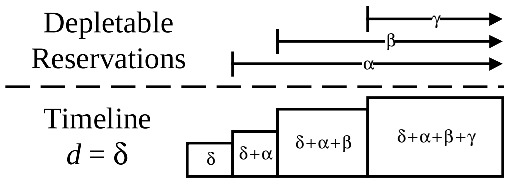

Figure 2 Effect of depletable reservations

Non-Depeletable Reservation: consists of a start- and end-time (s, e, as above) and a value (v). v is an integer representing the amount of capacity to be used by the reservation. The value of any point on such a timeline is the sum of all reservations that include the time-point in their temporal extents, as well as the default value. See Figure 3.

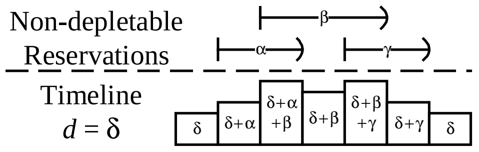

Figure 3 Effect of non-depletable reservations

State Reservation: we address two types of state reservations: changers and users.

A changer reservation consists of a start-time (*s*, the end-time is ignored) and a value (*v*). *v* is a symbol as described for state timelines. The value of any point on such a timeline is the value of the most recent changer reservation. If two or more changers are simultaneous to different values, the resultant value is invalid.

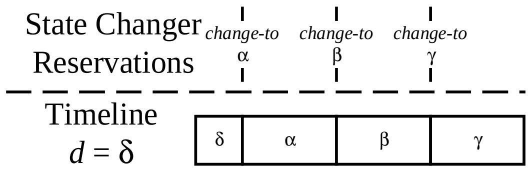

Figure 4 Effect of state changer reservations

A user reservation consists of a start- and end-time (*s*, *e*) and a value (*v*). *v* is a symbol as described for state timelines. A user reservation constrains the timeline such that for all points during its temporal extent, the value of the timeline is the same as *v*. Note that user reservations have no effect on the value of the timeline.

Conflict: a conflict is a violation of any constraints. These include: 1) over or under subscriptions, where a resource timeline value lies outside of its allowed range, 2) state transition violations, where a state timeline value is immediately followed by a value for which no corresponding transition arc exists in its transition graph, 3) state usage violations, where a state timeline value differs from a user reservation constraining the timeline during the temporal extent of the reservation.

Solution Description

Our approach to computing the valid intervals (*I*) is as follows:

1) gather all reservations of all activities, 2) partition these according to timeline, 3) compute the valid intervals for each partition (*P*), 4) translate and intersect the valid intervals. We focus on step 3. We know that if the reservations in *P* do not interact, then we can compute valid intervals for *P* by computing those for each individual reservation in *P*. This is accomplished by first computing each set of intervals I_r for each individual reservation *r* in *P*. We then translate I_r by the difference between the start-time of the reservation and the earliest start-time in *P*, and intersecting all of the resultant intervals.

The key point of our approach: transforming *P* into a set of non-interacting reservations *P'* is one way of making the computation of valid intervals tractable.

Even though reservations in *P* interact, the reservations in *P'* do not, thus we can compute valid intervals using simple translation and intersection of the valid intervals for each individual reservation in the *P'*.

Generating *P'* for non-depletable reservations is very straightforward. Simply create a new set of reservations that do not overlap temporally but cover the same temporal extent and represent the same values as those in *P*. See Figure 5. Note that the reservations in *P'* bear a similarity with the timeline, as in Figure 3. Also, note that the semantics of reservations in *P'* are the same as those in *P*. As we shall soon see, this is not always the case for other types of timelines.

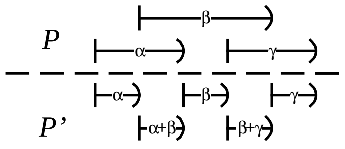

Figure 5 Non-depletable reservation transformation

Generating *P'* for depletable reservations is a little more interesting. First, we create a new set of non-depeletable reservations that represent the changes in the value of the timeline along the temporal extent of the of the depletable reservations in *P*. Finally, we add a single depletable reservation at the end of the collection of non-depletable reservations that represents the overall effect of the collection of reservations. See Figure 6. Note that the semantics for reservations in *P'* may differ from those in *P*.

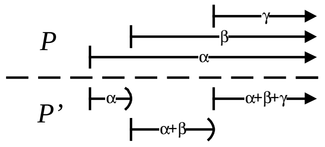

Figure 6 Depletable reservation transformation

Generating *P'* for state reservations is the main thrust of this paper. To accomplish the decomposition of a set of interacting reservations to a set of non-interacting reservations, we rely on a new crop of semantics. We will examine the following cases: 1) users only, 2) changers only, and 3) mixed reservations.

Users Only— Consider the case where *P* contains only user reservations. Trivially, if any two reservations in P overlap temporally and are of differing values, no possible non-conflicting set of intervals exists. Otherwise, simply intersecting the valid intervals for each individual reservation is adequate.

Changers Only— If P only contains changer reservations, problems due to interactions arise. Trivially, if any two reservations in P are simultaneous and are of differing values, no possible non-conflicting set of intervals exists. Otherwise, consider the case of a single changer reservation $r = (s, v)$. When looking for valid placements, we need to ensure that: 1) it isn't placed concurrently with a conflicting (non-equal valued) changer reservation already scheduled, 2) the transition from the value of the closest previous changer (or from the default value, in the case that no changer precedes it) to v is allowed, and 3) the transition from v to the closest subsequent changer's value is allowed. We decompose this single reservation into three separate reservations: 1) a *no-changers-or-equal* reservation to handle the case of simultaneous changers, 2) an *allow-change-to* reservation to handle the case where r is the first reservation in P, and 3) an *allow-change-from* reservation where r is the last reservation in P. See Figure 7.

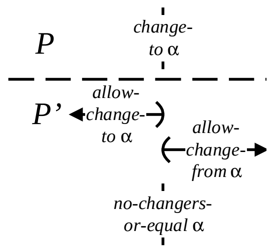

Figure 7 Transformation for a single state changer

But, what if we include more than one changer in P? Obviously, we require a *no-changers-or-equal* reservation for each reservation in P. We require an *allow-change-to* reservation for the first reservation in P, and an *allow-change-from* reservation for the last reservation in P. For each subsequent pair of changers $r_1 = (s_1, v_1)$ and $r_2 = (s_2, v_2)$, we require a reservation that represents the implied requirements of each changer reservation. We consider two cases: 1) the transition (v_1, v_2) is allowed, and 2) the transition (v_1, v_2) is not allowed. (Note: the terminology "transition (v_1, v_2) is allowed" simply means that $(v_1, v_2) \in E$, where $G = (V, E)$ is the transition graph for the timeline in question.)

For the first case, we need to ensure that either no changer intercedes or that the proper transitions are allowed. Consider the earliest interceding changer $i_e = (s_e, v_e)$ and the latest interceding changer $i_l = (s_l, v_l)$ (which may in fact be the same reservation.) We must ensure that both transitions (v_1, v_e) and (v_l, v_2) are allowed. We use two reservations: one for each case. These are: 1) a *no-changers-or-allow-first* reservation and 2) a *no-changers-or-allow-last* reservation. See Figure 8.

For the second case (the transition (v_1, v_2) is not allowed), we need to ensure that at least one interceding changer exists, and that the proper transitions are allowed, as described in the previous paragraph. So, we use two reservations: 1) a *need-a-changer-that-allows-first* and a *need-a-changer-that-allows-last* reservation. See Figure 9.

Finally, we must consider user reservations already scheduled. In this case, either the user must have the same value as the first changer, or an already scheduled changer must intercede the first changer and the scheduled user. We call this a *users-match-or-hidden* reservation. These are included in Figure 8 and Figure 9.

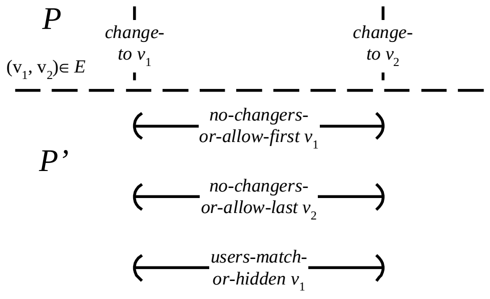

Figure 8 Allowed transition changer pair transformation

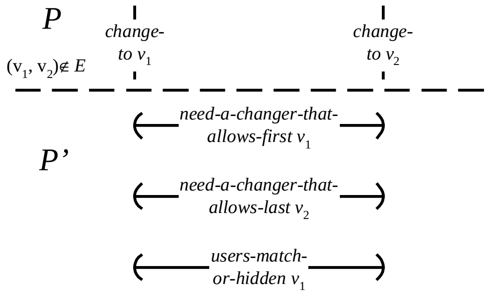

Figure 9 Disallowed transition changer pair transformation

Mixed Reservations— If P contains both changer and user reservations, potential interactions abound. Trivially, if a changer and user in P coincide (the start-time of the changer is contained within the temporal extent of the user) but are of differing values, no possible non-conflicting set of intervals exists. But, consider the case where all user reservations occur before any of the changer reservations. In this case, we can compute the valid intervals by computing the intervals as with **Users Only** and **Changers Only**, and intersecting. This is because no interactions can occur between users and changers. Trouble arises when users come after changers. In this case, the closest previous changer might help the situation by changing to the value of the user, or hinder it by changing to a different value. We consider each case in turn.

Consider the closest previous changer $r_c = (s_c, v_c)$ to a user $r_u = (s_u, v_u)$ where $v_c = v_u$. In this case, we either need no interceding changers or we need the latest interceding changer to change to v_u (i.e. match). We dub this a *none-or-last-changer-match* reservation.

But, consider the closest previous changer $r_c = (s_c, v_c)$ to a user $r_u = (s_u, v_u)$ where $v_c \neq v_u$. In this case, we need an interceding changer to change to v_u (i.e. match). We dub this a *need-last-changer-to-match* reservation.

For both cases, we require a *no-changers-or-match* reservation that ensures either no changers exist during the temporal extent of the user or the value of the changer equals v_u (i.e. matches). It is important to note that interceding users do not affect the validity of these reservations. See Figure 10.

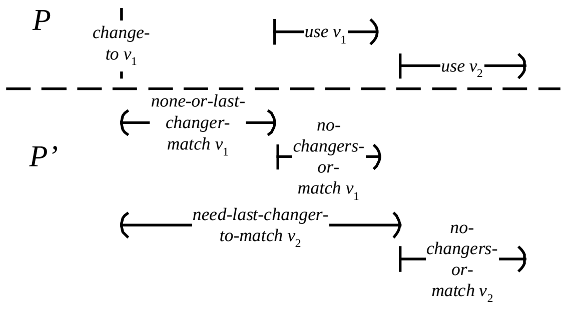

Figure 10 Equal and non-equal changer-user pair transformation

We now have our complete set of reservations. Valid intervals for each can be computed in time that is proportional to the number of reservations already scheduled. The number of reservations in *P'* is a constant factor more than those in *P*, making the overall asymptotic complexity of our technique equivalent to naïve computation, or roughly proportional to |*P*|· the number of reservations already scheduled.

Empirical Evaluation

We now describe an empirical comparison of an aggregate search technique using our informed approach for determining valid placements for collections of activities to the same search technique using a naïve approach. We evaluate two aspects of our algorithm: 1) quality in terms of speed and accuracy, and 2) efficacy in terms of conflict reduction, both in scheduling and combined planning and scheduling.

In our empirical analysis we use five models (and corresponding problem set generators): 1) the VTLI (valid timeline intervals) domain— a synthetic model designed to have inter-activity interactions, 2) the EO1 spacecraft operations domain, 3) the Rocky-7 Mars rover operations domain, 4) the DATA-CHASER shuttle payload operations domain, and 5) the New Millennium Space Technology Four landed operations domain.

Within each model and corresponding problem set, we generate random problems that include a background set of fixed activities and a number of movable activity groups. The activity groups are placed randomly. The goal is to minimize the number of conflicts in the schedule by performing planning and scheduling operations.

To solve each problem, we use the ASPEN (Automated Scheduling and Planning Environment) system using an "iterative repair" algorithm, which classifies conflicts and attacks them each individually (Fukunaga, A., Rabideau, G., Chien, S., Yan, D. 1997). Conflicts occur when a plan constraint has been violated; this constraint could be temporal or involve a resource or state timeline. Conflicts are resolved by performing one or more schedule modifications such as moving, adding, or deleting activities. The iterative repair algorithm continues until no conflicts remain in the schedule, or a timeout has expired.

The scheduler entertains non-conflicting placements when moving activity groups. In the *control* trials the scheduler does so using the naïve algorithm for computing valid placements. In the *experiment* trials the scheduler is using the informed method to compute valid placements. In all cases for each domain, both trials are using the same set of heuristics at all other choice-points (e.g., selection of a conflict or activity group to attempt to repair, where to place within computed valid intervals, etc.). Note that simple (non-aggregate) operations are available in both real domains, although they are of limited comparative utility. Using only non-aggregate operations, the problems are intractable within reasonable time bounds because the distance in terms of sub-optimal moves from one local optima to the next is $O(n)$ and the space to be searched is $O(m^n)$ where n is the number of activities in a movable collection and m is the number of possible locations given by a naïve calculation of legal intervals for an individual activity, e.g. in the EO1 domain, n ranges from 23 to 56; in the Rover domain, n ranges from 8 to 17. Note that naïve calculations for valid intervals for an individual activity are adequate for that activity.

We now briefly describe each domain including information on the types of activities and resources modeled, what the activity groups are, and how they are interdependent.

VTLI Domain

The VTLI domain consists of a color state variable and a charge resource. Color may be *red*, *purple*, or *blue*, and allows any transition except *red*-to-*blue* and *blue*-to-*red*. Charge is allowed to be an integer between 0 and 25, defaulting to 0. There are two types of activity groups for aggregation: color and charge. Each group consists of four activities: a color-activity may be either a user-of or changer-to any of the allowed colors; a charge-activity may use from -22 to 22 of charge. Because members of the same activity group all use the same timeline, there will be many intra-group interactions. For example, one member may change the color to red while a later member needs purple. This requires placing the pair with a transition to purple between them. Charge members may overlap, hence reserving the sum of their values during the overlapping period.

A VTLI problem instance includes random, fixed profiles for the timelines. Each timeline has 60 fixed reservations with values chosen from *red*, *purple*, or *blue* for color and from integers between 0 and 24 for charge. The values are chosen randomly, but without introducing inconsistencies with the model (e.g., illegal transitions). Each problem also includes ten movable groups of four activities with equal chance of each group being a color or charge group. We preclude intra-group reservations which are contradictory (e.g., overlapping members requiring different color states) but inter-group interactions may make problems unsolvable for a given fixed profile. The groups are initially placed at random times within the planning horizon.

EO1 Domain

The EO1 domain models the operations of the New Millennium Earth Observer 1 operations for a two-day horizon (Sherwood, R., Govindjee, A., Yan, D., Rabideau, G., Chien, S., Fukunaga, A. 1998). It consists of 14 resources, 10 state variables and total of 38 different activity types. Several activity groups correspond to activities necessary to perform different types of instrument observations and calibrations. The activity groups range in size from 23 to 56 activities, many of which have interactions. For example, taking an image of the earth requires fixing the solar array drive to avoid blurred images. The high-level observation activity group includes both commands to fix the SAD and take the image.

Each EO1 problem instance includes a randomly generated, fixed profile that represents typical weather and instrument pattern. Each problem also includes 8 randomly placed instrument requests for observations and calibrations.

Rocky-7 Domain

The Rocky-7 Mars rover domain models operations of a prototype rover for a typical Martian day (Rabideau, G., Chien, S., Backes, P., Chalfant, G., and Tso, K. 1999). It consists of 14 shared resources, 7 state variables and 25 activity types. Resources and states include cameras (front, rear, mast), mast, shovel, spectrometer, solar array, battery, and RAM. There are four activity groups that correspond to different types of science experiments: imaging a target, digging at a location, collecting a spectrometer reading from target, and taking a panoramic image from a location. Activity group size ranges from 8 to 17 activities. Members in activity groups have positive resource interactions, e.g. opening the aperture for the camera enables subsequently taking a picture. Activity groups also have negative interactions, e.g. several member activities using the onboard buffer. Rover problems are constructed by generating four experiments and randomly generating parameters for the experiments (such as target locations).

New Millennium Space Technology Four Landed Operations Domain

The ST4 domain models the landed operations of a spacecraft designed to land on a comet and return a sample to earth. This model has 6 shared resources, 6 state variables, and 22 activity types. Resources and states include battery level, bus power, communications, orbiter-in-view, drill location, drill state, oven states for a primary and backup oven state, camera state, and RAM. There are two activity groups that correspond to different types of experiments: 1) mining and analyzing a sample, 2) taking a picture. Activity group sizes range from 5 to 10. As in the rover domain, activities interact positively and negatively.

Each ST4 problem instance includes a randomly generated, fixed profile that represents communications visibility to the orbiting spacecraft. Each problem also includes five mining and two picture experiments (each randomly placed.)

DATA-CHASER Domain

The DCAPS domain models operations of a shuttle science payload that flew onboard Space Shuttle Flight STS-85 in August 1997. It consists of 19 shared resources, 25 state variables, and 70 activity types. Resources and states include shuttle orientation, contamination state, 3 scientific instruments (doors, relays, heaters, etc.), several RAM buffers, tape storage, power (for all instruments/devices), and downlink availability. There is one type of activity group corresponding to one experiment for each of the 3 scientific instruments. This activity group consists of 23 activities. As with the other domains, activities in this activity group interact positively and negatively.

Each DCAPS problem instance includes a randomly generated, fixed profile that represents shuttle orientation and contamination state. The number of randomly placed experiments ranges from 2 to 20 based on the fixed profile for the given problem instance.

Quality Assessment

To assess the quality of the algorithm, we directly compare the accuracy and speed of the informed search mechanism to the naïve intersection approach and a random placement approach. We assess the accuracy of the competing approaches by comparing the intervals for legal placement that each algorithm returns to the correct intervals. We assess the speed of the three approaches by measuring the CPU time taken by each algorithm to compute its intervals.

Accuracy is a desirable property in any algorithm to determine “good” placements for aggregated activities. Ideally, a legal interval generator would return exactly those times that are legal placements for the member activity. This would mean that the algorithm would be sound (e.g., all times in the interval returned would be legal) and complete (e.g., all legal times would be returned by the algorithm). Table 1 shows the results of this evaluation on the ST4, EO1, Rover, and DCAPS domains. Because the informed method is complete and sound, it returns all of the correct interval(s) and no incorrect intervals. However, the naïve intersection method has both false positive (soundness) and false negative (completeness errors). As the data shows, it tends to miss the majority of the legal interval (by failing to recognize positive interactions between activities in the collection).

	EO1	Rover	DCAPS	ST4
informed	$0/_{14870}$	$0/_{10030}$	$0/_{5100}$	$0/_{96890}$
naïve	$14880/_{10}$	$4250/_{5780}$	$5700/_{3600}$	$51690/_{45220}$
random	2742250/ 2757120	$16101/_{17104}$	$2100/_{7200}$	360150/ 457040

Table 1 Errors in Calculated Intervals ($^{\text{avg. errors}}/_{\text{avg. nterval size}}$)

Speed is another desirable property of a legal times algorithm. Ideally, a legal interval generator would take very little CPU resources to compute. Table 2 shows the average CPU time taken by each algorithm to compute its estimation of the legal intervals. Because the random algorithm simply returns the whole interval it takes effectively zero time. We also observe that the informed algorithm takes more time than the naïve algorithm. This not surprising as it must first transform the basic state and resource reservations into non-interacting reservations, then compute legal intervals for each, and then intersect them. The naïve approach need only perform the latter two steps.

	EO1	Rover	DCAPS	ST4
informed	1.8925	.025	.3528	.0315
naïve	.1506	.025	.1090	.0300
random	0	0	0	0

Table 2 Average time to compute intervals, in seconds

Efficacy Assessment

We assess the efficacy of our algorithm in terms of conflict reduction. We compare the number of conflicts reduced for scheduling operations, and the overall effect this has on solving combined planning/scheduling problems.

In terms of scheduling, ideally, placing an aggregate at a time recommended by a legal times algorithm should result in an improved schedule (i.e. one with fewer conflicts). Table 3 shows the average reduction in the number of conflicts in the schedule after placement of the aggregate. Note that because having an unplaced activity is a conflict, by default each algorithm gets a score of *n* if there are *n* activities in the aggregate just for placing the activity (i.e. in the absence of any state or resource conflicts introduced or removed). We see that the informed algorithm strictly outperforms both naïve and random placement.

	EO1	Rover	DCAPS	ST4
informed	21.2878	1.9042	25.2549	5.800
naïve	19.9978	1.8339	13.8416	0.9997
random	19.2978	-0.7837	21.4041	3.0785

Table 3 Effectiveness through conflict reduction

To assess the effect that the algorithm has in solving planning/scheduling problems, we examine the number of conflicts over time (in terms of CPU usage) and the total number of problems solved for each domain. If superior, the informed search algorithm should result in faster reduction of conflicts and more problems being completely solved.

We generate twenty random problems for each domain and run ASPEN with twenty different random seeds for each combination of problem and technique. Note that we do not guarantee that the problems are solvable.

We evaluate the performance of our technique versus the performance of the naïve technique in terms of the number of iterations to solve conflicts, amount of time to solve conflicts, and the total number of problems solved for each domain.

For the VTLI domain, our informed technique is slightly slower than the naïve technique per iteration, but performs better than the naïve technique in terms of the number of conflicts solved. This slowdown is expected in that transformation of interacting state reservations leads to a constant factor more non-interacting reservations.

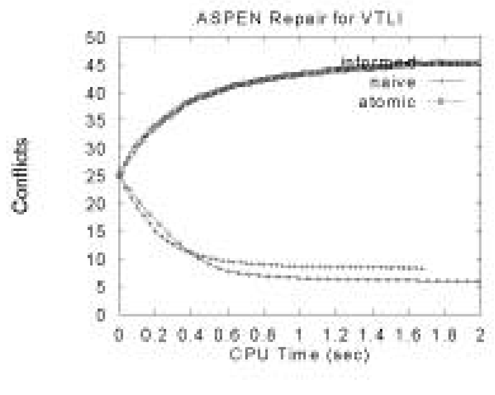

Figure 11 Conflicts over time for our synthetic domain

For the EO1 operations domain, the naïve technique and informed technique perform similarly at first. This is because a number of the conflicts do not involve interacting reservations and hence the naïve technique can

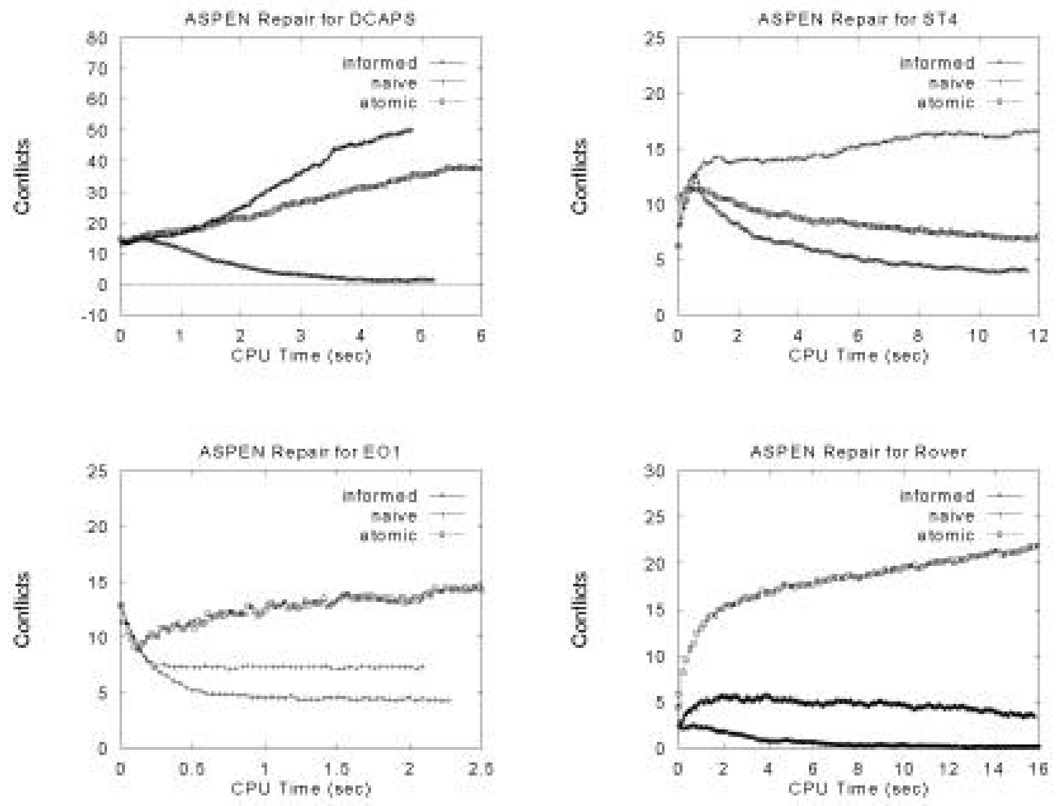

Figure 12 Conflicts over time for real domains

solve them. However, many of the conflicts involve interacting reservations. Because the informed technique correctly handles these interactions, it is able to solve these conflicts. Thus in the longer term the informed algorithm is able to solve more conflicts in the schedule.

For the Rocky-7 Mars rover operations domain, the informed technique appears to strictly dominate the naïve technique. Interestingly, conflict count rises before it falls for both algorithms. This is due to the added planning necessary to solve conflicts. Adding activities leads to more conflicts initially, but eventually leads to solutions.

For the New Millennium ST4 Landed Operations domain, the informed technique strictly outperforms the naïve technique. Conflict count rises before it falls as in the rover domain and for the same reason, except the algorithm employing the naïve technique never recovers. Many of the activities in a group interact, therefore the naïve technique often makes mistakes in recommending placements for activity groups. Because of this faulty advice, repair using the naïve approach actually increases the number of conflicts in the schedule.

For the DCAPS domain, the informed technique strictly outperforms the naïve technique. In this domain, almost all of the activities in a group interact, leading to similar consequences as the ST4 domain.

In terms of number of problems solved, we observe that ASPEN employing the informed scheduling technique is able to completely solve (i.e., remove all conflicts) more problems than ASPEN employing the naïve approach in all five domains (Table 4).

	VTLI	EO1	Rover	DCAPS	ST4	total
informed	84/400	149/400	390/400	387/400	243/400	1253/ 2000
naïve	4/400	60/400	243/400	1/400	0/400	308/ 2000

Table 4 Problems Solved

These empirical results imply that aggregate reasoning is effective in synthetic and real domains, both in terms of number of constraint violations repaired and in terms of overall time to reach a desired solution quality, as long as we use an informed scheduling function. We have computed the statistical confidence that the average final number of conflicts using the informed search method is less than the final number of conflicts using the naïve method. For the VTLI, EO1, ST4, and DCAPS domains, this confidence is greater than 99.9%. For the Rover domain, this confidence is greater than 98%.

Discussion and Conclusions

There are a number of related systems that perform both planning and scheduling. IxTeT (Laborie, P., Ghallab, M. 1995) uses least-commitment approach to sharable resources that does not fix timepoints for its resource and state usages.

HSTS (Muscettola, N. 1993) enforces a total order on timepoints affecting common shared states and resources, allowing more temporal flexibility. We believe that our technique is applicable in this case at a greater computational expense (while still being polynomial), and future research should address this issue.

Both IxTeT and HSTS are less committed representations than our grounded time representation and this flexibility incurs a greater computational expense to detect and/or resolve conflicts.

O-PLAN (Drabble, B., and Tate A. 1984) also deals with state and resource constraints. O-PLAN's resource reasoning uses optimistic and pessimistic resource bounds to efficiently guide its resource analysis when times are not yet grounded. Like ASPEN, O-PLAN also allows multiple constraint managers which would enable it to perform general reasoning when times are unconstrained and more efficient reasoning in the case where all timepoints are grounded (also enabling aggregate informed search as described in this paper).

SIPE-2 (Wilkins, D., 1998) handles depletable/non-depletable resource and state constraints as planning variables using constraint posting and reasons at the same level of commitment as IxTeT.

(Cesta, Oddi S., and Smith S. 1998) apply constraint-posting techniques to satisfy multi-capacitated resource problems at the same level of commitment. Depletable/non-depletable resource constraints are easily transformed to multi-capacitated resource constraints. None of these systems generally consider aggregate operations in their search space.

Conclusion and Acknowledgements

This paper has described the use of informed transformation techniques to improve the efficiency of scheduling sets of interdependent activities. We describe our algorithm for processing interacting state and resource requirements of a cluster of interdependent activities into a set of independent requirements and use these to search for placements for the activity set. We show empirically that our informed search method outperforms the alternative approach of searching for legal placements on both synthetic problems and problems from space exploration domains. Finally, we wish to thank Stephen Smith of Carnegie Mellon University and Richard Korf of the University of California, Los Angeles for their helpful suggestions concerning this paper.

References

Cesta, Oddi S., and Smith S. 1998. "Profile-Based Algorithms to Solve Multiple Capacitated Metric Scheduling Problems." *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, Pittsburgh, Pennsylvania, 1998, pp. 214-223.

Dechter, R., Meiri I., and Pearl J. 1991. "Temporal Constraint Networks," *Artificial Intelligence*, 49, 1991, pp 61-95.

Drabble, B., and Tate A. 1984. "Use of Optimistic and Pessimistic Resource Profiles to Inform Search in an Activity Based Planner," Proc. AIPS94.

Estlin, T., Chien, S., and Wang, X. 1997. "An Argument for an Integrated Hierarchical Task Network and Operator-based Approach to Planning," in *Recent Advances in AI Planning*, S. Steel and R. Alami (eds.), Lecture Notes in Artificial Intelligence, Springer—Verlag, 1997, pp. 182-194.

Fukunaga, A., Rabideau, G., Chien, S., Yan, D. 1997. "Towards an Application Framework for Automated Planning and Scheduling," *Proceedings of the 1997 International Symposium on Artificial Intelligence, Robotics and Automation for Space*, Tokyo, Japan, July 1997.

Kautz, H., and Selman B. 1996. "Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search." *AAAI-96.*

Laborie, P., Ghallab, M. 1995. "Planning with Sharable Resource Constraints," Proc. IJCAI-95, 1643-1649

Lin, S. and Kernighan, B. 1973. "An Effective Heuristic for the Traveling Salesman Problem," *Operations Research* Vol. 21, 1973.

Muscettola, N. 1993. "HSTS: Integrating Planning and Scheduling." *Intelligent Scheduling*. Morgan Kaufmann, March 1993.

Rabideau, G., Chien, S., Backes, P., Chalfant, G., and Tso, K. 1999. "A Step Towards an Autonomous Planetary Rover," *Space Technology and Applications International Forum*, Albuquerque, NM, February 1999.

Sherwood, R., Govindjee, A., Yan, D., Rabideau, G., Chien, S., Fukunaga, A. 1998. "Using ASPEN to Automate EO-1 Activity Planning," Proceedings of the 1998 IEEE Aerospace Conference, Aspen, CO, March 1998.

Tate A., Drabble, B., and Dalton, J. 1996. "O-Plan: a Knowledge-based planner and its application to Logistics," *Advanced Planning Technology, Technological*

Achievements of the ARPA/RL Planning Initiative, AAAI Press, 1996, pp. 259-266.

Vidal, V., and Regnier, P., 1999. “Total Order Planning is More Efficient than we Thought.” *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, AAAI Press, 1999, pp. 591-596.

Wilkins, D., 1998. “Using the SIPE-2 Planning System: A Manual for Version 4.22.” SRI International Artificial Intelligence Center, Menlo Park, CA, November 1998.

Zweben, M., Daun, B., Davis, E., and Deale, M., 1994. “Scheduling and Rescheduling with Iterative Repair,” in *Intelligent Scheduling*, Morgan Kaufman, San Francisco, 1994.